M11 SEGURETAT I ALTA DISPONIBILITAT

Administració de Sistemes Informàtics en Xarxa
IES de l'Ebre

UF4 – Alta Disponibilitat

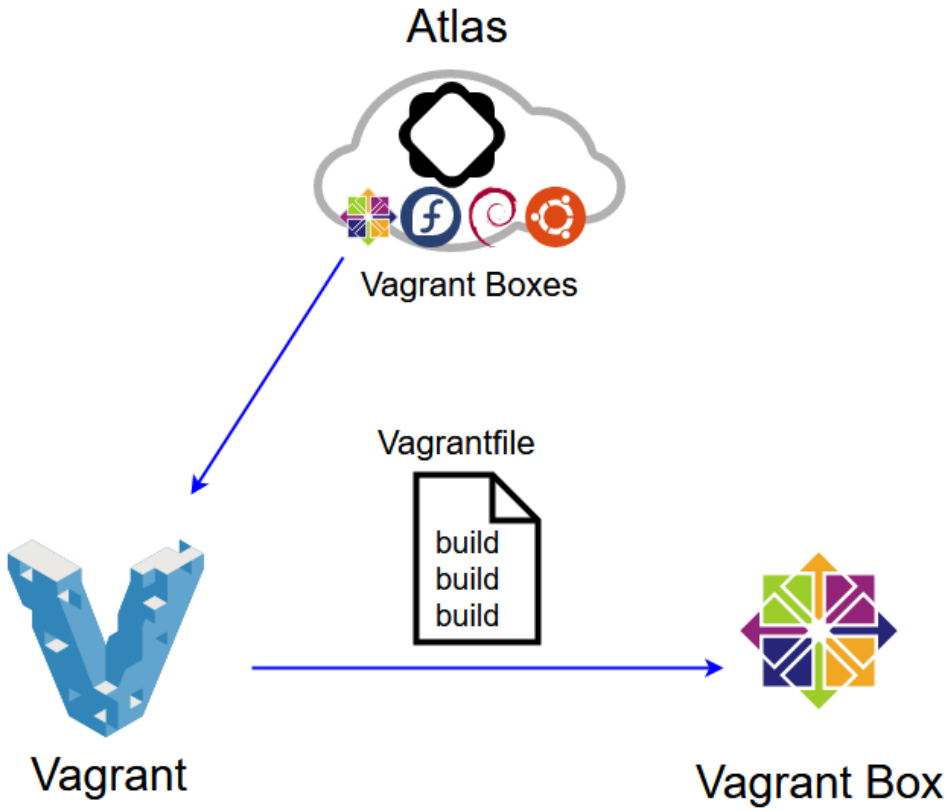
AUTOMATITZACIÓ – Vagrant + Ansible

Vagrant :: Què és?

- Vagrant és una eina que ens permet **crear escenaris virtuals** d'una forma molt senzilla i replicable.
- És un projecte de **codi obert** escrit en Ruby i **multiplataforma** que va néixer el 2010 de la mà de Mitchell Hashimoto, dos anys després es va llançar la primera versió estable i es va crear l'empresa HashiCorp per assegurar el desenvolupament i el suport de Vagrant.
- Orientat a l'ús per **desenvolupadors o entorns de producció** simples.
- Pot **integrar-se amb eines de gestió de la configuració**.
- Forma part del conjunt d'aplicacions utilitzades en "**infraestructura com a codi**".
- Github: https://github.com/hashicorp/vagrant

Vagrant :: Què és?

- **Problema**
  - Configurar escenaris a mà es tediós i provoca errors.
  - Un desenvolupador ha de centrar-se en el desenvolupament.
  - Els escenaris poden canviar.
- **Solució**
  - Distribuir de forma separada imatges "Netes" del SO i la configuració completa
  - Molt lleuger
  - Fàcil de modificar i redistribuir
  - Fàcilment integrable en el flux de treball "devops"
  - Molt útil per a programari lliure



Vagrant :: Ús bàsic

- Instal·lació de Vagrant
  - `sudo apt install Vagrant`
- Descarreguem una imatge (box) d'Ubuntu18
  - `vagrant box add ubuntu/bionic64`
- Creem un directori per a cada projecte, accedim i generem un fitxer amb la configuració per defecte:
  - `vagrant init -m ubuntu/bionic64`
- Iniciem la màquina, que utilitzarà la imatge descarregada per desplegar una nova màquina i l'arrencarà:
  - `vagrant up`
- Podem accedir a la màquina utilitzant la connexió per defecte (p 2222) que crea vagrant quan desplega una nova màquina:
  - `vagrant ssh`
- Podem comprovar en tot moment l'estat de la màquina amb:
  - `vagrant estatus`
- Podem aturar la màquina amb:
  - `vagrant halt`
- Si de nou aixequem la màquina, es reutilitzarà la màquina anteriorment creada i els canvis que s'haguessin efectuat en ella. Quan no vulguem tornar a utilitzar aquesta màquina la podem eliminar permanentment amb:
  - `vagrant destroy`

Vagrant :: Ús bàsic

- Guia ràpida de Vagrant
- https://www.busindre.com/guia_rapida_de_vagrant
- Gestió de Vagrantfiles
- https://picodotdev.github.io/blog-bitix/2019/05/crear-de-forma-sencilla-y-rapida-maquinas-virtuales-de-virtualbox-con-vagrant/

VAGRANT

Ansible :: Què és?

- Ansible **automatitza l'aprovisionament de programari**, la gestió de configuracions i el desplegament d'aplicacions. Està categoritzat com una **eina d'orquestració.** Permet als devops gestionar els seus servidors, configuracions i aplicacions d'una forma senzilla, robusta i paral·lela.

- Ansible gestiona els seus diferents nodes a través d'SSH i únicament requereix Python al controlador en el qual es vagi a executar per poder utilitzar-lo.

- **Utilitza YAML** per a descriure les accions a realitzar i les configuracions que s'han de propagar als diferents nodes.

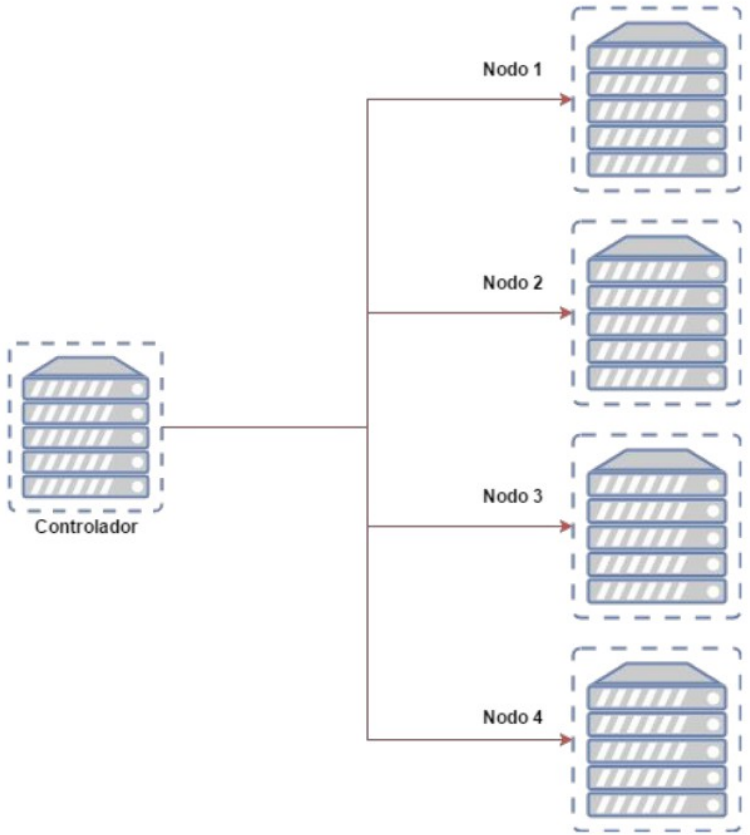
- El nom d'ansible és usat en la literatura de ciència ficció per descriure **un dispositiu de comunicació més ràpid que la llum**. El concepte és utilitzat a **El Joc d'Ender.**

- L'autor és **Michael DeHaan** i el repositori s'allotja a **Github**: https://github.com/ansible/ansible

- El projecte té una **comunitat gran i activa** al darrere.

Ansible :: Arquitectura

- Hi ha dos tipus de rols a ansible:
  - **Controlador**: és la màquina des de la qual comença l'orquestració.
  - **Node**: és gesti



ANSIBLE

Ansible :: Controlador

- El node controlador ha de tenir instal·lat tant ansible com python:

```
apt-add-repository ppa:ansible/ansible (si es necessari)
```

```
apt-get update
```

```
apt-get install ansible python
```

```
ansible --version
```

- És necessari crear un parell de claus amb l'usuari amb que llençarem les comandes ansible:

```
ssh-keygen (assegureu estar amb root per als nostres exemples)
```

- Ja només ens cal crear el nostre inventari a /etc/ansible/hosts

```
[servers]
server1 ansible_host=192.168.11.101
server2 ansible_host=192.168.11.102
```

- Podem comprovar la creació amb la comanda

```
ansible-inventory --list -y
```

ANSIBLE

Ansible :: Nodes

- En els nodes que seran orquestrats pel controlador no cal instal·lar cap programari addicional, però si assegurar-nos de que l'accés per ssh és possible, per això comprovarem que és permeti accés root (compte això no és recomanable) i es demani contrasenya:

```
sudo nano /etc/ssh/sshd_config
```

```
PermitRootLogin yes
```

```
#PasswordAuthentication no (ens assegurem que la línia estigui comentada)
```

```
service ssh restart
```

- També hem de tenir en compte que es necessari que l'usuari root tingui contrasenya.
- NOTA: realitzem la pràctica amb l'usuari root tot i que seria recomanable utilitzar un usuari creat per a l'ocasió amb els permisos necessaris per a realitzar només les gestions remotes requerides.
- Ara ja només ens cal tornar al nostre controlador, assegurar-nos de que tenim accés per ssh als nodes i copiar la clau del nostre usuari controlador als nodes per tal d'estalviar-nos tenir que introduir la contrasenya cada cop que accedim o llencem un ordre.

```
ssh-copy-id root@ip_node
```

ANSIBLE

Ansible :: Comandes AD-HOC

- En primer lloc comprovem la connectivitat amb els nodes:

```
ansible all -m ping -u root
```

- Anem a comprovar el hostname de tots dos nodes de forma paral·lela amb ansible:

```
ansible all -a "hostname"
192.168.33.11 | SUCCESS | rc = 0 >> node-one
192.168.33.12 | SUCCESS | rc = 0 >> node-two
```

- Tenen els nostres servidors prou espai d'emmagatzematge lliure?

```
ansible all -a “df -h”
```

- Memòria disponible

```
ansible all -a “free -m”
```

- Sincronització de data i hora. La manera més fàcil de fer-ho és utilitzar NTP (Network Time Protocol).

```
ansible all -a "date"
192.168.33.11 | SUCCESS | rc = 0 >> Sat juny 3 16:27:02 UTC 2021
192.168.33.12 | SUCCESS | rc = 0 >> Sat juny 3 16:27:02 UTC 2021
```

ANSIBLE